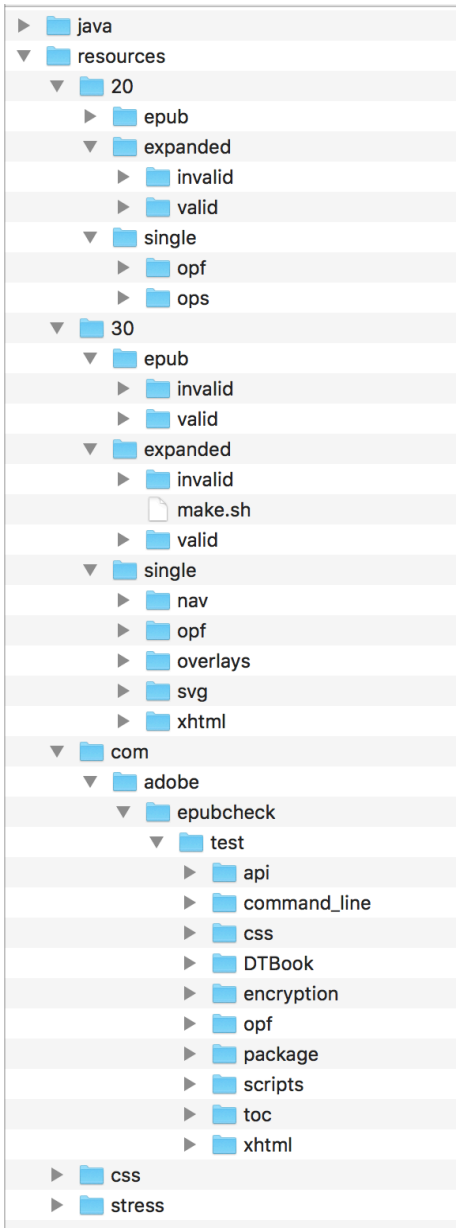
epubcheck Tests Directory Structure Proposal

Status Quo

The current directory structure within the repository is:



That's basically four horizontal slices through the specification – EPUB 2.x, EPUB 3.x, Common (the adobe files), and CSS – each with somewhat different takes on substructure.

Within these directories, there are various types of test files: packaged EPUBs, exploded EPUBs, and individual resources (images, CSS files, XHTML, etc.). These is also various test-runner code as well as expected result files. All in all, somewhat of a mishmash (reasonably) built up over time with input from various implementers.

Even though out of scope for this document, proposing a going forward common directory structure for all test files, it would seem as though coalescing around a common test file type would be a worthwhile organizational enhancement. It is suggested that this be a packaged EPUB file (xxx.epub), with each likely based on the template provided by Dave Cramer. Each EPUB file should be a standalone test, hopefully testing a single feature or failure. With this approach, test runner code could be minimized and could likely be scripted. The exception to this "all EPUB files" approach would likely want to be a few exploded EPUB directories as required to validate epubcheck's support for this input scheme.

The competing approach to the above (which Brady supports) is to favor exploded EPUBs over their zipped incarnation. This structure would have advantages in editability, change tracking, and versioning. Garth leans toward packaged EPUB's for reasons of "eating our own dogfood", transportability, and utility outside of this specific application. Regardless of the final decision on canonical test file format, the below proposed directory structure would be unchanged.

Consensus from 2017-07-06: unzipped files

Unit Tests

Above and below, only EPUB test files are considered – files used to verify that specific features are validating and that specific errors are detected. Also to be considered, and equally important, are unit tests for epubcheck itself. Unit tests will almost certainly be Java code (with some sort of resource input files) and should not be mixed in with EPUB test files. A "unitTests/" directory (likely with substructure) should exist beside the below proposed "testFiles/" directory and should contain the plethora of unit tests.

Proposed Test Files Directory Structure

Two high level organizational structures were considered: version based and specification based (with the current arrangement being some amalgamation of each). A version based structure requires historical knowledge in order to place or locate tests and may not provide a definitive location as particular features evolve over time and versions. Thus, a specification or functional based organization is prefered and proposed here.

The proposed top level directory structure is:

- testFiles/commandLineInterface
- testFiles/container
- testFiles/package
- testFiles/content

Leading lower case, long names, and camel caps, are proposed here, but that's driven by habit without particularly strong religion, and could be changed upon implementation as long as a single approach is used throughout the structure.

The first directory would be used for any tests required to validate epubcheck's command line interface, and would likely be the only directory containing any exploded EPUB files. The three subsequent directories contain tests for the ZIP archive, the package and its associated components, and content documents, respectively.

It's unclear that the first two directories need any substructure, and thus they would directly contain tests, likely with the canonical structure being:

- xxxx.epub
- xxxx-expectedResults.txt

With the first being the EPUB test file and the second being the file with the expected results. If it was decided to use exploded EPUB files as the canonical format, the ".epub" above would be ".dir" and the exploded contents would live below that.

Text is shown here for expected results, as a preference, but the current tests use a combination of text, JSON, and XML. Coalescing around a single format is important. Text would seem desired for ease of observation and test running, but the implementer could drive toward JSON or XML.

An initial (and extensible) substructure for the package tests would be:

- testFiles/package/opf
- testFiles/package/ncx
- testFiles/package/navigation
- testFiles/package/fallbacks

An initial (and extensible) substructure for the content document tests would be:

- testFiles/content/markup
- testFiles/content/images/bitmap
- testFiles/content/images/svg

- testFiles/content/scripting
- testFiles/content/css
- testFiles/content/mediaOverlay
- test/content/fixedLayout

Additional substructure could be added as required, but roughly the proposed level of hierarchical bifurcation is desired. “markup” is proposed as the first sub-directory name over “xhtml” to plan for a future including “html”, but the implementor could overrule this preference.